From Exotic to Mainstream

A 10-year Odyssey from Internet Speed to Boundary Spanning with Scrum

Baskerville, Richard; Pries-Heje, Jan; Madsen, Sabine

Published in:
Agile Software Development

Publication date:
2010

Document Version
Peer reviewed version

Citation for published version (APA):
Baskerville, R., Pries-Heje, J., & Madsen, S. (2010). From Exotic to Mainstream: A 10-year Odyssey from Internet Speed to Boundary Spanning with Scrum. In T. Dingsøyr, T. Dybå, & N. B. Moe (Eds.), *Agile Software Development: Current Research and Future Directions* (pp. 87-110). Springer Science+Business Media.

5 From Exotic to Mainstream: A 10-year Odyssey from Internet Speed to Boundary Spanning with Scrum

Richard Baskerville, Jan Pries-Heje, Sabine Madsen

Abstract. Based on four empirical studies conducted over a 10-year time period from 1999 to 2008 we investigate *how local software processes interact with global changes in the software development context*. In 1999 companies were developing software at high speed in a desperate rush to be first-to-market. In 2001 a new high speed/quick results development process had become established practice. In 2003 changes in the market created the need for a more balanced view on speed and quality, and in 2008 companies were successfully combining agile and plan driven approaches to achieve the benefits of both. The studies reveal a two-stage pattern in which dramatic changes in the market causes disruption of established practices, experimentation, and process adaptations followed by consolidation of lessons learnt into a new (and once again mature) software development process. Limitations, implications, and areas for future research are discussed.

5.1 Introduction

Over the course of the last ten years, agile software development has received much attention from both the practitioner and research community, first as a novelty and later as a development approach that has become widely used in practice (Dybå and Dingsøyr 2008). In this chapter we look at how software development in practice has changed over this ten year time period. More specifically we compare and contrast the practices of Internet speed and agile software development at four different points in time: When the internet was booming in 1999; during the peak of the "dot.com" boom in 2000-2001; just after this economy collapsed in 2002-2003; and most recently in 2008. For simplicity, the four studies and points in time are here after referred to as study one from 1999, study two from 2001, study three from 2003, and study four from 2008.

Right before the beginning of the millennium the Internet was being adopted faster than nearly any other technology. It took 30 years (1920-1950) for the telephone to reach a 60% penetration in USA. It took 15 years for computers to reach a 60% penetration. But it only took 2 years for the Internet to reach 60% penetration (Atlanta_Constitution 2001). In 1999 we therefore compared the growth of the Internet to an exploding bomb, and we called this phenomenon the "e-bomb" (Baskerville and Pries-Heje 2001).

At this point in time, in 1999, we carried out an interview study in three Danish companies. The study revealed that the then present notion of software development methodology was changing. In fact we found that the lack of methodology in its traditional form was characteristic. Instead of methodology, time pressure and requirements ambiguity was found to be at the core.

Two years later, in 2001, we did a comprehensive study in US. Ten companies that themselves claimed to be working at Internet speed were interviewed. Data analysis identified three major factors that influenced Internet software development processes: demand for rush to market, a different kind of market environment, and the lack of experience developing software for the Internet. Further we identified a new software development process used within a unique and enthusiastic development culture.

In 2003, after the dot.com bubble had burst we interviewed in the same companies. Fundamental changes in the economic conditions now affected the resources available for Internet software development and expectations had changed dramatically, resulting in three outcomes. First, the IT economy underwent a major upheaval as revenue fell, productivity rose, and budgets were slashed. Second, business expectations changed. Rather than an unbridled obsession with fast software delivery, customers demanded both speed *and* quality. Third, the economy drove an emphasis on the business case for software projects, and the concerns of the project managers changed to encompass the value to the enterprise, including development of more complex, mission critical software systems.

After the publication of our internet studies agile methods, and especially Scrum (Rising and Janoff 2000) and eXtreme Programming (XP) (Beck 2000; Beck and Fowler 2001; Jeffries et al. 2001), became popular in practice. However, the ideal settings for the use of agile methods versus more traditional methods were much discussed. Boehm (2002) has for example speculated on what constitutes the agile 'home ground', defined as the application area in which agile ISD has its special strengths and performs best given the project characteristics. Boehm and Turner (2004) have also suggested a radar diagram to characterize software projects and thereby obtain a recommendation on whether to use agile or 'disciplined' methods. Cockburn (2002) suggested a framework where one axis was number of people and the other was criticality (life at risk) of defects. He described an ideal setting (with up to 20 people and no serious money or life at risk) as the 'sweet spot' in which agile methods were preferable.

Some years later, in 2008, more and more companies were adopting Scrum in both Denmark and US (the two places where we live and work and thus have the closest contact to). It also looked as if Scrum was being used outside the 'sweet spot'. Therefore we identified and conducted interviews in three Danish companies that were using Scrum near the edge of its suggested application area. This usage was occurring in larger, sometimes geographically distributed teams and with essential money at risk. All three companies studied were successful in organizing the use of both Scrum and a plan-driven approach to achieve the benefits of both, namely the ability to respond quickly to change *and* the alignment of long-term plans and on-going activities.

In this chapter we provide a historical overview over the changes that the practical phenomenon of agile software development has gone through with regard to the aspects of time, application area, scope, and organization from 1999 to 2008. The research methodology and results address questions of *how local software processes interacts with global changes in the software development context.*

We have organized the remainder of this chapter in the following way. First we describe our research methodology, which is anchored in grounded theory tech-

niques. Then we summarize the individual story line that proceeds from each of the four study periods. Lastly we conclude with a discussion of the overall story line that covers the 10-year time span.

5.2 Research Methodology

We have undertaken four phases of research, using Grounded Theory (GT) as our research methodology. GT is a qualitative research methodology that takes its name from the practice of discovering theory that is grounded in data. This research methodology does not begin with a theory, and then seek proof; rather it starts with an area of study and allows the relevant theory to emerge from that area (Strauss and Corbin 1998). The research outcome is grounded theories that are inductively discovered by careful collection and analysis of qualitative, empirical data. Use of GT in IT research is exemplified by a landmark paper by Orlikowski (1993) on CASE tools and organizational change, as well as explorations on software requirements (Urquhart 1997, 2000). GT is best used in research where one has relatively "uncharted land", which for example was the case with the notion of 'Internet speed'.

Our research questions in the 1999 and 2001 studies revolved around the concept of Internet speed: What does it mean? Is there something one could distinguish as "Internet speed development"? How is it different from or similar to traditional development? In 2003, we continued to ask about Internet speed, but were more focused on what had changed from the boom to the bust. In 2008 agile development had become widely diffused and successfully used, also beyond the application area initially recommended by the agile method authors. Our research interest therefore centered on the question of how agile development, and more specifically the agile method, Scrum, was used in projects at or well beyond the edge of its original sweet spot and why this seemed to work.

For all four phases of research, we have collected our data via semi-structured interviews. The interview guide was structured around the following topics:

1. The firm, and its' products and services
2. The interviewee
3. Projects in the organization – from start to end
4. Development model used?
5. Internet time / Agile development – What does it mean to you?
6. The development process itself
7. Talent, training, learning, and knowledge
8. Transfer of knowledge
9. The biggest problem / Greatest challenge?

Each interview lasted approx. 1-1½ hour, relevant documents were collected, and observation notes were recorded (e.g., about the use of open- or closed space offices; and the general impression of the pace, atmosphere, and 'tone' of the work place).

For data analysis we have applied the three coding procedures of GT (Strauss and Corbin 1998) called open, axial and selective coding.

The goal of open coding is to reveal the core ideas found in the data. Open coding involves two tasks. The first task is labeling phenomena. This task involves decomposing observations into discrete incidents or ideas. Each discrete incident or idea receives a name or label that represents the phenomenon. These names represent a concept inherent in the observation. The second task is discovering categories. Categorizing is the process of finding related phenomena or common concepts and themes in the accumulated data in order to group them under joint headings, thus identifying categories and sub-categories of data.

The purpose of axial coding is to develop a deeper understanding of how the identified categories are related. Axial coding also involves two tasks. The first task connects categories in terms of a sequence of relationships. For example, a causal condition or a consequence can connect two categories, or a category and a sub-category. The second task turns back to the data for validation of the relationships. This return gives rise to the discovery and specification of the differences and similarities among and within the categories.

Selective coding involves the integration of the categories that have been developed to form the initial theoretical framework. First, a story line is generated or made explicit. A story is simply a descriptive narrative about the central phenomenon of study; the story line is the conceptualization of this story (abstracting). The story line becomes the core category, which is related to all the categories found during axial coding, thereby validating these relationships, and elaborating the categories into a theoretical expression that explains the phenomena observed.

5.2.1 Study One: Interview Study in Denmark

The first phase of our research aimed at exploring the influence of working on Internet time (Cusumano and Yoffie 2000). One could say that we were testing the hypothesis that working on Internet time would have to cause some changes in the way software development work was organized. But beyond this no hypotheses were pre-formulated and tested.

We interviewed in three Danish companies. Two of the companies were new to the authors and the third was a company we had visited over a period of time for a longitudinal study. The main facts about the three companies are given in Table 1, and further details can be found in Baskerville and Pries-Heje (2001).

Name (Pseudonym)	What offered?, When founded?, Which size?	Number of people interviewed and their organizational roles
NewWays	Develops custom-tailored Internet products for major customers internationally. Founded in the mid 1990s. 50 employees when interviewed.	4 people interviewed: One project manager, a development manager and two developers.
ProfWeb	Develops custom-tailored Internet and	2 people interviewed: A development

	Intranet products interfacing with large existing databases.	manager and a developer.
	Founded in the early 1990s.	
	40 employees when interviewed.	
AlfaWeb	A general web-based product sold on the market as a standard product for e-commerce.	2 people interviewed: The CEO and a development manager.
	Founded in the late 1990s.	
	12 employees when interviewed.	

Table 1. Facts about the three companies (study one).

5.2.2 Study Two: Interview Study in USA

The second phase of our research involved ten detailed case studies of Internet software development companies in two major U.S. metropolitan areas. The firms ranged in size from 10 employees to more than 300,000 employees and covered different industries in the private and public sectors including: financial services, insurance, business and consulting services, courier services, travel, media, utilities, and government services. Some of the firms were Internet start-ups while others were "brick and mortar" companies with newly established Internet development units.

The objective was to understand whether software development for the Internet differs from traditional software development. This phase identified the practices used for Internet software development and explored the role of quality in fast-cycle development environments (Baskerville and Pries-Heje 2002). Further details on this study are given in Baskerville et al. (2003)

5.2.3 Study Three: A Follow-up Study

Another round of interviews in the same companies as in phase 2 was conducted two years later. Only five of the original ten companies (from 2001) remained in business or were available to participate in the study. A brief description of each firm is provided in Table 2, and further details are available in Pries-Heje et al (2005).

Name (Pseudonym)	What offered?, When founded?, Which size?	Number of people interviewed in each round and their organizational roles
Calliope	Offers forecasting tools for energy and communications industry.	2001: 3 people interviewed: VP Operations, Project Manager, Software Developer.
	Founded in the mid 1990s.	
	20 employees when interviewed.	2003: Not interviewed.
Clio	Low-price health care and utilities for	2001: Six people interviewed: President

	groups of customers. Founded in the late 1990s. 35 employees when interviewed.	& CEO, VP Technology Operations, Director of Marketing Research, Chief Information Officer, two developers. 2003: Not interviewed.
Deca	Develops and markets a platform of E-business software modules that allow users more control when doing business online. Founded in the late 1990s. Approx. 10 employees when interviewed.	2001: Not interviewed. 2003: Four people interviewed: CEO, developer, QA specialist and marketing manager.
Erato	Offers to help Brick & Mortar companies get online. Founded in the late 1990s. 55 employees when interviewed.	2001: Four people interviewed: Director, Chief Financial Officer, Chief Operations Officer, and developer. 2003: Not interviewed
Euterpe	Film and Television Industry. Offers high-tech tools online. Founded in the mid 1990s. 80 employees when interviewed.	2001: Four people interviewed: Project managers, marketing specialists, senior web developers. 2003: Not interviewed
Melpomene	Carries out personnel administration for other companies online. Founded in the mid 1990s. More than 100 employees when interviewed.	2001: Seven people interviewed: Project managers, process improvers, architects, user interface designers, web developers. 2003: 6 of 7 people interviewed. Process improvement person had left company.
Polyhymnia	Offers online services for transport and tourist industry. Founded in the early 1990s. More than 1000 employees when interviewed.	2001: Six people interviewed: Senior managers, Project managers, QA manager, lead developers, web developer. 2003: Seven people interviewed: Same distribution of roles as in 2001.
Terpsichore	Offers industrial insurance online. Founded in the 1930s. More than 10000 employees when interviewed.	2001: Three people interviewed: Human Resources Manager, Internet site manager and Internet site developer. 2003: Not interviewed.
Thalia	Online service for transport and logistics industry. Founded in the 1930s. More than 100000 employees when interviewed.	2001: Six people interviewed: CIO, Senior manager, project managers, architects, senior developers, web developers. 2003: Three of the six people interviewed: CIO, senior and project manager.
Urania	Business-to-business communication. Founded in the 1980s. More than 100,000 employees when interviewed.	2001: Six people interviewed: Senior manager, Project managers, quality assurance manager, QA specialist, Web developers. 2003: Six people interviewed. Same roles. But only three were the same people.

Table 2. Facts about the ten companies (study two and three).

5.2.4 Study Four: Scrum Interview Study in Denmark

The fourth round of interviews was conducted for the purpose of exploring how Scrum was used in projects characterized by larger and geographically distributed teams concerned with the development of business and life critical software. Three Danish companies were selected as relevant sites for data collection as their IT projects exhibited these characteristics (See Table 3). The case companies had from one year to two and a half years of experience with the use of Scrum, with SuperSystem being the most experienced.

Name (Pseudonym)	What offered?, When founded?, Which size?	Number of people interviewed in each round and their organizational roles
GlobeRiver	Develops engineering products with built-in intelligence (software). Founded in 1940s. 500 employees in R&D function worldwide when interviewed; of this 25 in a company-owned development house in India (Developers and Scrum masters).	3 people interviewed: a Danish Scrum master, a Danish Facilitator, and an Indian Scrum master.
SuperSystem	Develops software for the military, the banking industry, hospitals, etc. Founded in 1980s. Approx. 400 employees when interviewed.	4 people interviewed: a Lead Developer, a Scrum master, the manager of the internal software process improvement (SPI) department, and the person officially in charge of implementing Scrum in the company.
DareYou	An off- and online gaming company; works with several suppliers located in different places and countries to develop the online games. Founded in 1940s. Approx. 250 employees when interviewed.	2 people interviewed: The Project manager and the Product owner.

Table 3. Facts about the three companies (study four).

The results of the four phases of research are presented below in the form of four grounded theories. The theories cover several levels of analysis, namely the market, the portfolio, the project, and the team level. However, many of our respondents were operating at the project and team level (project managers and developers). We have therefore been able to collect more detailed data, conduct more thorough analyses, and develop more robust theories about these two levels.

5.3 Study One Results: Racing the E-bomb

In the first study we noted ten properties of a new methodology for "e" development (Baskerville and Pries-Heje 2001). Each of these properties is briefly described below, along with examples of how these properties are manifested in the cases. We also describe the chain of causal links that we discovered among these properties, which helps explain why this particular set of properties has come to characterize Internet time development (an early manifestation of agility). These properties and the causal chain are depicted graphically in Figure 1.

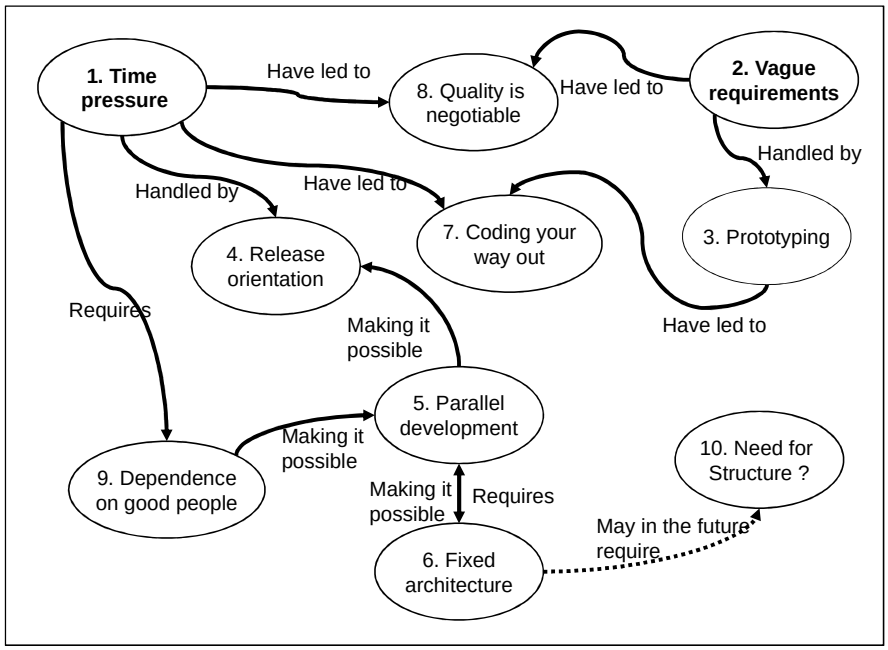

Fig. 1. Results from the first study.

Time pressure. We found time pressure to be a condition permeating software development in the three companies we studied. First-to-market is the central, defining high-priority goal of Internet time development. Minimizing time-to-market from concept to customer is an all-consuming activity and achievement of this goal drives almost all other elements of the methodology. This goal is not new in business (Smith and Reinertsen 1995) nor in software development (Cusumano and Selby 1995; Iansiti and McCormack 1997). However, the degree to which it influenced systems development had not yet been recognized when we conducted this study.

Vague requirements. An inability to pre-define system requirements is the central, defining constraint of Internet time development. The requirements specification has traditionally been the heart of systems development. However, Internet time development accepts a starting point in which the requirements are per-

mitted to persist in near or full ambiguity. For example a project manager at NewWays said, “Often a project starts without a requirements specification. …companies come to us and say: We believe there is a treasure buried in the World Wide Web. … we want something new.”

Prototyping. The idea of using prototypes seems to be widespread and permeating both early and late work in development projects. For example ProfWeb describes their use of prototypes as being part of their core competence. The R&D manager said: “We live from being technologically in front of our competitors, and from being able to visualise more far-reaching and wide-ranging solutions for our customers than our competitors are able to.”

Release orientation. The vague requirements are not just something we see in the beginning of a project. In fact it continues throughout the development process. One consequence is what we have named a “release orientation”. Software systems are produced in a series of ever more refined and extensive versions of the product; and each release contains bug-fixes and new features. These maturing product cycles characterize Internet software development in which competition demands significant product and feature changes every few months (Cusumano and Selby 1995). This release orientation helps relieve some of the time pressure because if a feature does not make it for the contemporary release, it can simply be postponed to the following release, which is never very far behind.

Parallel development. The release orientation demands a fast cycle time that is impossible to meet in a serial process. Parallel development processes therefore flourish, meaning that a number of activities take place at the same time. Products and releases therefore have to be designed and coordinated for parallel development, another aspect common to Internet software development (Cusumano and Selby 1995). For example, NewWays projects typically have a duration of 2-3 months. A sequential, waterfall-like model is seen as much too slow. Instead NewWays have several parallel development processes running at the same time.

Fixed architecture. To make parallel development possible, it is necessary to have some means for dividing the work. In all three cases we found that this has led to the use of a fixed three-tier architecture. At NewWays the development manager describes it in the following way: “Architecture is important to NewWays. Typically an application has three layers: At the bottom you have a database with content; in the middle you have the business logic; and at the top you have the HTML generating logic, typically written in Visual Basic Scripts”. The architecture is used as an important coordination mechanism to divide the work in the project. It is explained that: “Typically the graphical person is drawing something in PhotoShop which the HTML person then can cut up and put into tags,” says one developer and another continues, “Which means that we are released from worrying about presentation and can concentrate on the heavy things” [i.e. the business logic and the database].

Coding your way out. The short time frame allowed for developing applications also introduces a coding focus or even hacking: “You have to accept that hacks are being made, that you don’t have time to think systematically, and that you don’t reuse because of the time pressure” (NewWays).

Quality is negotiable. Three different ways of looking and talking about quality have appeared over the last 20 years (Crosby 1980). One school of thought fo-

cuses on fulfilment of customer expectations. Another way of thinking emphasizes measurable product attributes and conformance to requirements. The third approach is process oriented and assumes that a good development process will lead to quality. The three resulting kinds of quality can be named expectation-based, product-based, and process-based quality.

As a consequence of both time pressure and vague requirements we found that both product-based and process-based seemed to be ignored. Moreover, customers and users seemed to expect low quality. We decided to call this phenomena negotiable quality.

ProfWeb was for example struggling with quality. They knew it was not good enough and they had started thinking about what to do: "We collect a Test Group for every project. At least that is the plan for the future, but right now we are running the pumps, not financially, but we are very busy … I have a capacity planning system and the UNIX department is booked 4 months ahead" (ProfWeb). Thus time pressure is a cause of the negotiable quality.

Dependence on good people. Time pressure is also the primary reason why good people are in high demand. As one of the founders of ProfWeb phrased it: "I believe the largest bottleneck we have is to get enough qualified employees". However, not all kinds of IT people were in high demand. Traditional analysts were not in as high demand as the technical people who were close to the code: "I also realised that the job market is such that I could find 25 new consultants tomorrow but I wouldn't be able to find two new programmers" (ProfWeb).

Need for new kinds of structure. An issue that is closely related to methodology and to a number of issues we have addressed above is structure. We have not been able to establish a solid causal relationship, but we have indications that seem to reveal that the older and larger the organization and/or the customers the larger the need for structure. For example, AlfaWeb, which only had existed for half a year when we interviewed them, was not feeling any need for structure. The CEO explained: "I believe it is the informality but also the lack of formal structures. If people have to close-knit a framework to work in they might cut down on creativity" (AlfaWeb). In contrast, NewWays, which had existed for two years and had 50 employees, had started creating some structures, and had started using a number of object-oriented techniques.

5.4 Study Two Results: A New Software Development Process

In the second study we identified three major categories of observations that were causing a change, and three major categories that were resulting from the changing causes (Baskerville et al. 2003). Key findings are that Internet software development is different from traditional development and that the case companies are getting good at developing software at Internet speed by using an increasingly established set of practices that facilitate quick results, i.e. by using a new (agile) software process.

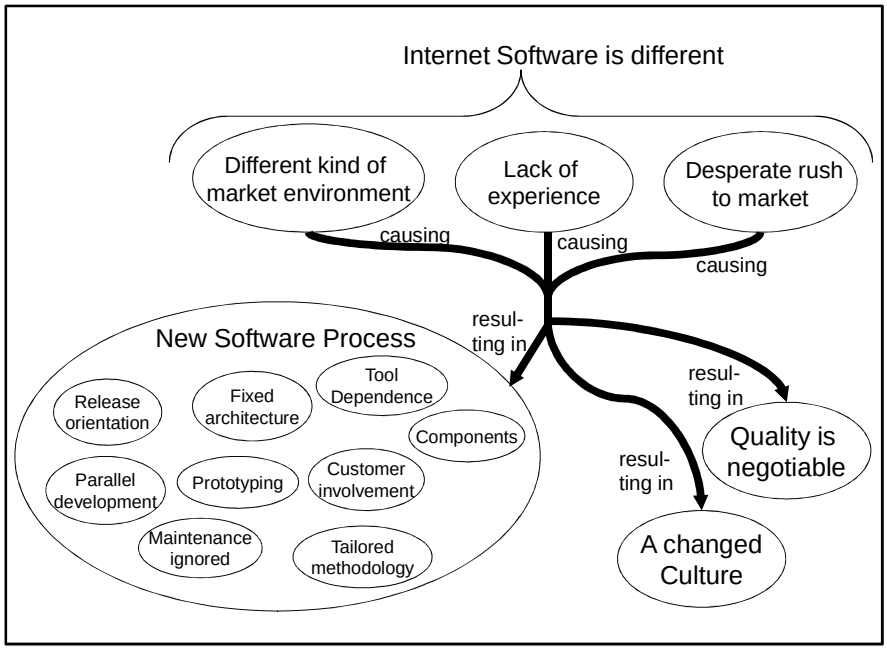

Fig. 2. Results from the second study.

A different kind of market environment. The Internet created a unique platform and marketplace for software products - one that was flexible in terms of requirements and quality. Requirements and quality were negotiable from release-to-release in a market-oriented process where competition and pragmatics were allowed to intervene to limit the scope of features in each release.

Lack of experience. The interviewees reported that there were too few knowledgeable and experienced developers who understood the new technology, changing market conditions, and who could meet the need for speed. A manager from Melpomene told us that "lots of people [in our organisation] came from more corporate environments where it took forever to get things out the door." Much of this prior experience was a hindrance rather than a benefit in the new environment. The shortage of experienced professionals made the marketplace for developers tight and expensive, and created development organisations that lacked sufficient experience and expertise.

Desperate rush to market. In all the case companies they explained that Internet software development was driven by a desperate rush to market. "Time-to-market…Bigger, faster, better. Everything is very rush, rush, rush" (Polyhymnia).

Quality is negotiable. Many quality factors were not as critical in Internet speed development as they were in traditional software development. Customers and users appreciated quick results and were willing to defer a certain amount of reliability and performance until later releases. And developers were willing to rebuild badly designed or coded features later when the deferment ran out. "It is dif-

ferent working at Internet speed. Compressed cycles mean that quality suffers. With speed we are sending poorer quality out the door" (Polyhymnia).

A changed culture. We found that Internet software development organizations had a distinct culture that appreciated informal structure, smaller teams, and diverse team compositions. Moreover, there seemed to be a tight bond among Internet software developers, a sense of belonging with others who shared the same values. "We are not 9 to 5 people down here. We are more dynamic … There is a lot more excitement and enthusiasm here" (Thalia).

A new software process. At the project level, we identified nine distinct characteristics (see Table 4). Although no single characteristic was unique to the new development process, the collection of characteristics was distinctive, aimed at producing quick results, and remarkably common in the case companies.

Characteristic of the new software process	Description and examples
Parallel development	To achieve high speed we found that companies compressed development into a time frame where only overlapping, parallel development could meet the demands.
Release orientation	"People have a perception of Internet speed. They expect it. So we've had to scope our delivery or deliver a smaller set of features. Thereby releasing more often", said a manager from Euterpe. Clio said: "Development cycles last from 2 to 15 days… timing is important. Features that cannot be completed in time can slip from one release to the next. The fast cycle time softens the penalty from slipping a feature."
Tool dependence	Urania estimated that "fifty percent of development is already taken care of by tools we use such as iplanet or websphere. The APIs to these tools gives a lot of functionality." Many Internet software development organizations made heavy use of development tools and environments that could speed up the design and coding process. Furthermore new tools also helped to create well modularized and architected systems.
Customer involvement	When requirements are fuzzy it helps having close access to customers. Thus intimately involving customers to cope with evolving and unstable requirements was typical. We also found that customers were often co-located with the development team, and participated closely in all phases of development. Most projects relied on such involvement rather than a formalized requirements management process.
Prototyping	Instead of using formal requirements documents, most projects used prototyping as a way to communicate with their customers to validate and refine requirements. Customers would describe the basic functionality for new or changed features and these were quickly prototyped for demonstration and experimentation. "We are supposed to have a full [requirements and design document] but a lot of programmers use the prototype and go back and forth to check, or go back and ask: what was this supposed to do" (Melpomene).
Criticality of architecture	A well-planned architecture enable each release to be developed with some similarity. A three-layer architecture was common: (1) Database layer, (2) Business logic layer, the detailed processing code, and (3) User interface layer.

Components based development and reuse	Internet speed can be achieved by software assembled with as many reusable components as possible, rather than crafted from scratch. “Internet speed needs reuse. We need to take components and to know how to put them together” (Thalia).
Maintenance ignored	The short life span of Internet software meant that maintenance often was not given serious consideration. “Products are not documented. No design document, no requirements specification. The person who did it is gone. It takes much longer time. Often we can start from scratch. It leads to a throw away mentality”(Polyhymnia).
Tailored methodology	The processes and methods used in Internet software development varied considerably depending on the composition of the project team and the nature of the product. "We have an overall methodology. But we have to tailor processes for individual teams”(Urania). Just "enough process to be effective", added Euterpe.

Table 4. Nine characteristics of the new software process (study two).

5.5 Study Three Results: Balancing Speed and Quality

Three major changes took place from our second to our third study, i.e. in just two years (Pries-Heje et al. 2005). First, quality was no longer being treated as a disadvantaged stepchild. Speed and quality had to be balanced for companies to survive. Second, the unending supply of money that characterized the dot com boom had dried up. Third, good people were no longer in such short supply.

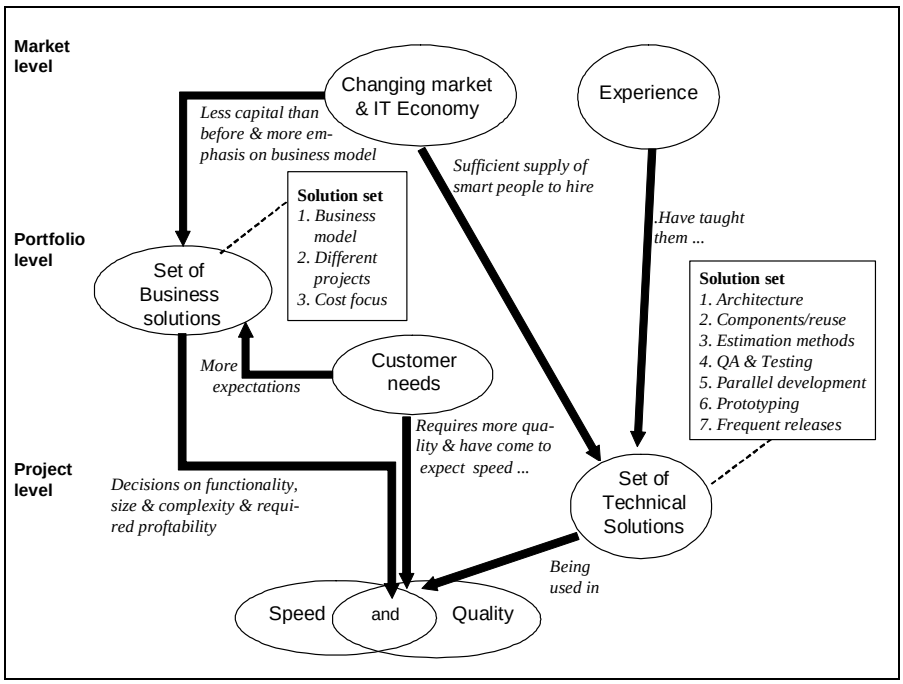

Fig. 3. Results from the third study.

5.5.1 The Market Level

At the market level two things had changed. **The changing market and IT economy** were having visible impact on the firms that we visited. The companies were still under pressure to develop and deliver software at Internet speed, but they were operating with significantly lower capital than before, and were no longer finding it difficult to hire or retain the talent they needed. Most of the companies described their states as either holding back or cutting down on their employees. Thus, changes in the IT economy were especially noticeable with respect to personnel and staffing. Moreover, disappearing venture capital as well as tight budgets forced the companies to focus on business value and costs. What is new is the employment of a set of business solutions at the portfolio level, carefully coordinated with use of a set of technical solutions at the project level.

Also contributing to the state of Internet software development at this point in time is the hard-won **experience** gained during both the dot com boom and bust. Whereas development staff at the companies we visited in our second study expressed boundless energy, excitement, and some confusion about what they were supposed to do and how, in our third study a more mature and reflective perspective was evident. One member at Melpomene suggests that caution contributed to

the company's survival: "I am critical of the phrase 'Internet speed', of the dotcom craze and demise. At that time, there was excitement. But all the successful practices were successful because they were good practices. We made conservative decisions and are still here." Thus, significant learning has taken place in these companies, across a wide range of business and technical topics. A member of Thalia sums up the learning in the following way: "Products are getting better due to more experienced developers."

5.5.2 The Portfolio Level

Sets of business solutions. The changing market and IT economy resulted in increased attention to the need for business models and a new, or increased focus on costs. "What has changed? We don't waste time on things that don't generate revenue" (Thalia).

As opposed to the days of abundant resources where risky projects even with faint hopes of success were undertaken, organizations were now much less willing to fund projects that did not have a clear business case. "We have to balance the need to do things fast and [the] desire to do it right – you need to have a business case." (Urania). A manager at Thalia also explains that his "Product is expected to generate revenue. That is different than before. Now we need to make a business case for each project". This situation encouraged project managers to clearly articulate the rationale for their projects, position them appropriately in alignment with organizational needs and requirements and in addition, market them to relevant decision makers.

All of the companies were refining their identities and offerings, but the challenges were being tackled in different ways. Some companies harkened back to more conventional business models, recognizing that "Success from now on depends on being a software and service company rather than an Internet company" (Melpomene), while others were forming partnerships with external (development) organizations.

Customer needs. The voice of the customer was still very much present, expressed through product strategy concerns, relationships, and ongoing contact. "The speed hasn't changed. If anything it gets faster and faster as customer expectations grow…[the biggest challenge] is meeting your customer's expectations" (Urania). Customer needs remained a challenge to discern and satisfy and at the same time customer expectations - for speed *and* quality - were significantly higher than in the second study.

5.5.3 The Project Level

The project level categories manifested themselves as a set of technical solutions. Of the seven categories in this solution set, five were somewhat similar to the process elements in study two. These five similar elements are: a standard architecture, the (re)use of components, parallel development, prototyping, and fre-

quent releases. Two new elements appeared: estimation and the improvement and involvement of quality assurance (QA) and testing. The two new elements are described in Table 5.

Characteristics of the technical solution set	Description and Examples
Estimation	A major difference between our second and third study was the recognition of the need for good estimation methods to track and improve performance. The organizations that had been involved in internet software development for a few years declared that they were more mature in their estimation of effort and schedule - “we know what it is like to develop in this environment" (Polyhymnia).
Improve and involve QA and Testing	With markets and products maturing, quality was getting more important. QA and testing was now seen as important aspects of software development. Due to the time-constrained environment, the need for more efficient QA was stressed. “If I look at a project time line, a lot of it is in QA testing. We need to improve and automate and create scripts to drive that down” (Thalia).

Table 5. Two distinctive characteristics of the technical solution set (third study).

Speed and Quality. Individuals in all the case companies commented explicitly on the struggle to balance speed and quality. “E-speed and e-haste are just normal now. Now you just know that you have to go that way and balance for quality” (Urania). The need for speed appeared to be as constant in our third study as it was in the second study. The customers had gotten accustomed to high speed development and were expecting it in every project. Quality, however, was viewed as having greater importance than previously seen. Quality was associated with number of defects, customer satisfaction, and overall success. Our interviewees explained that “If you don’t follow your processes, or do your documentation, that is not quality” (Urania) and that “They'll forget that you're late but they won't forget if it's bad” (Polyhymnia). Thus, at this point in time all three types of (product, process, and expectation-based) quality had become important.

5.6 Study Four Results: Boundary Spanning with Scrum

In our two studies conducted before the Dotcom bust, software development for the Internet was characterized by time pressure. In the third study changes at the market level led to a more balanced view on speed and quality; business value and costs.

In the forth study, we examined three companies that were using an agile method (Scrum) for some parts of their software development process and a more traditional approach for other parts of their development efforts. The case compa-

nies were motivated to use Scrum by an internal drive to achieve the benefits of an agile approach. The following benefits were highlighted as particularly important:

1. A closer contact with and immediate feedback from the customer.
2. Increased developer commitment and feelings of ownership.
3. The energy released from being able to focus on quick results.

At the same time, and due to the size and distributed nature of the team work as well as the criticality of the software, the interviewees stressed the need for alignment. Alignment is described as necessary to ensure that the work carried out by the Scrum teams is in line with the overall scope document, budget, and project plan. The work should also align with the major milestones of the project and broader company-prescribed methods and standards (e.g., CMMi). The wish for energy and agility and the need for overview and alignment has led all three case companies to employ 'a mixed strategy' (Abrahamsson et al. 2009) where they combine the relatively recently adopted agile approach with more well-established plan-driven ways of working (see Figure 4).

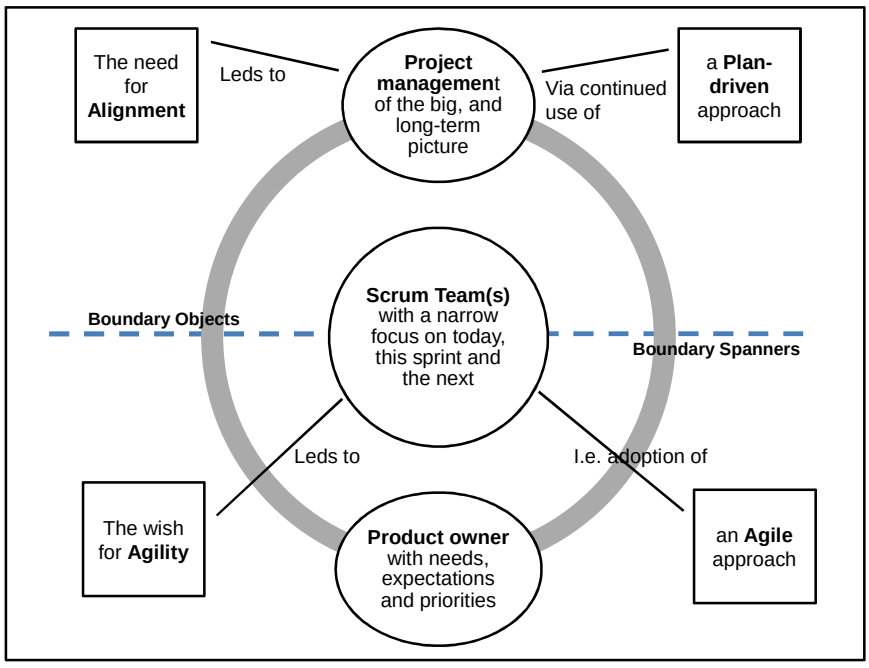

Fig. 4. Results from the forth study.

The three companies are very clear about how they organize to achieve the benefits of both the agile and the plan-driven approach. They explain that "We carry out project planning and management at several levels." (SuperSystem). Thus, in all three cases, the Scrum team(s) and master(s) are allowed to have a narrow focus on 'today, tomorrow, this sprint, and the next', while a project man-

ager has the responsibility for the project and the overall alignment of plans and people, i.e. for the big, and long-term picture. This division of work is necessary because "Scrum does not help with the overall, long-term planning of the project…you need to have an additional layer of project management. Otherwise it is not possible to coordinate and oversee a project with a deadline 1½ to 2 years into the future" (SuperSystem).

As the companies have chosen to organize for both agility and alignment certain Scrum elements and key people come to play a boundary spanning role that facilitates the sharing and negotiation of knowledge between several intersecting, but distinct social worlds (Levina and Vaast 2005; Star and Griesemer 1989).

5.6.1 Agility

The companies explain that they are "…true to Scrum at the team level" (SuperSystem) and that they "…use Scrum more or less 'by the book'" (GlobeRiver). Thus, Scrum helps the Scrum team conduct the work that is a part of their social world, namely the coordination and performance of tasks (related to analysis, design, development, test, integration, and release) as well as the monitoring of progress, risk, and quality for the current sprint. Moreover, Scrum plays an important role in providing a number of boundary objects that mediate the interaction that takes place between the Scrum team and the customer organization. For the product owner the prioritized user stories constitute functionality that will be a part of the next deliverable, and for the developers the user stories are (also) tasks that have to be carried out during the particular sprint. At the same time, the stories and the software allow the team members and the product owner to communicate, share knowledge, and create new meaning across the boundaries of their different worlds.

5.6.2 Alignment

In all three case companies software development was previously conducted in accordance with a sequential, document-oriented development model, and a plan-driven approach continues to be used after the introduction of Scrum. However, the plan-driven approach is now separated from the development teams and activities, and used by an appointed project manager for overseeing the project as a whole. It is explained that "…surrounding the team's work and the burn-down chart is the overall project plan, including milestones, broad-level estimates, a mapping from milestones to sprints, and a plan over external releases, as well as a risk analysis. The board-level estimates and the mapping from milestones to sprints help validate if the project and its scope can be achieved within the time frame and the budget, but the details of the sprints are not specified in these plans. That is the responsibility of the team" (SuperSystem).

DareYou also reported that, even though they are the customer organization, they consider themselves responsible for the project and its success. Consequently,

they also operate with an appointed 'traditional' project manager, as well as project management tools and documents such as a written project vision, budget, overall project plan and some up-front specification of requirements. In this case, the customer's project manager is the boundary spanner who keeps the project and its plans and participants aligned, and together with the product owner she is heavily involved in and well-informed about the actual quality and progress of the suppliers' development efforts.

In GlobeRiver the overall management and responsibility for the development of new engineering products resides with the R&D department. Thus, the Indian Scrum teams carry out the software development within the frame of large, business critical projects that involve many internal departments and external suppliers and which are managed in accordance with a traditional plan driven approach. It is very important that the software meets the deadlines in the road map for the product development project as a whole and that it is in line with the requirements and quality in the specification. In this setting, the R&D project manager is also the product owner and responsible for prioritizing the already specified functionality, which the Scrum teams then develops during a number of sprints. Moreover, a third role, a Facilitator, has been introduced. The Facilitator, located in Denmark, "…is the main point-of-contact between the Danish product owner and the Indian teams and follows-up on progress and impediments on a weekly basis, or more if needed…" (GlobeRiver). The Facilitators serve as boundary spanners who use certain information objects (such as, e.g., the road map, product backlog, burn-down chart, and impediments list) to keep the Indian Scrum team informed about requirements, priorities, and deadlines and the Danish R&D manager up to date about progress, quality, and risks. In this way, the Facilitator plays an important role for the translation of information between the agile and the plan driven worlds. This in turn allows the Indian software developers and the Danish R&D personnel to operate almost completely according to their own goals and work practices.

In sum, in all three case companies, the combined use of Scrum and a plan-driven approach has been organized so that the involved agile and plan-driven communities-of-practice (Cox 2005) can work largely in keeping with their own goals, information needs, and methods. Consequently, the translation of information and negotiation of new meaning across the different intersecting worlds is necessary. To this end, certain information elements (i.e. the overall project plans and burn-down charts) function as boundary objects, while the Scrum masters in SuperSystem, the Project manager in DareYou, and the Facilitator in GlobeRiver have boundary spanning roles, which they are fully aware of.

5.7 Discussion and Conclusion

Over the ten year time span, learning has been generated from each of the four studies. Much of this learning might be characterized as detailed and "keen insight" that is created within each of the studies and which does not bear well in

brief summaries. However, it is possible to consider the learning that arises from the accumulation of insights across the four studies. A limitation of this approach lies in its assumption that the four studies provide serial episodes. Because the studies involve differing subject organizations and individuals, we cannot rule out the alternative explanation that the consistencies in the data sets are accidental.

With this caveat in mind, we can summarize and interpret the collection of four studies as follows. The central story line in the first study brought the changing landscape of software development into sharp focus. In this study, it appeared that the two main sources driving software development were incredible time pressures coupled with unknown and changing requirements. These two primary causal factors arose as the context for software development changed due to the emergence of the E-economy.

The central story line in the second study embraced a new software process that was common across the respondents. This process included customer involvement, parallel development, a release orientation, etc. The components of the new process were present in the first study, but had become more or less established practice in the second study (Baskerville and Pries-Heje 2004).

In the third study, the story shifted again, but this time away from software development in a local sense. This story line focuses instead on changing economics and the role of software in formulating business solutions and generating revenue. A balancing game arises in which business and technical factors are brought together by high speed software projects.

Finally, in the fourth study, the central story line shifts once more, but the focus falls back on software process. In this study, we find organizations which are not exactly integrating agile and planned software processes; rather they are operating these two different ways of working consistently within separate boundaries. Work is flowing across the boundaries to enable the organizations to harvest the benefits they require from each of the deployed software processes (agile and planned).Thus, boundary objects and spanners play a key role in this story.

An interesting aspect of these study settings is the historically repeating two-stage pattern where the story line first centers on a changing context and then on the software process, almost as a maturation in response to early adaptation to changes in the context (see Figure 5).

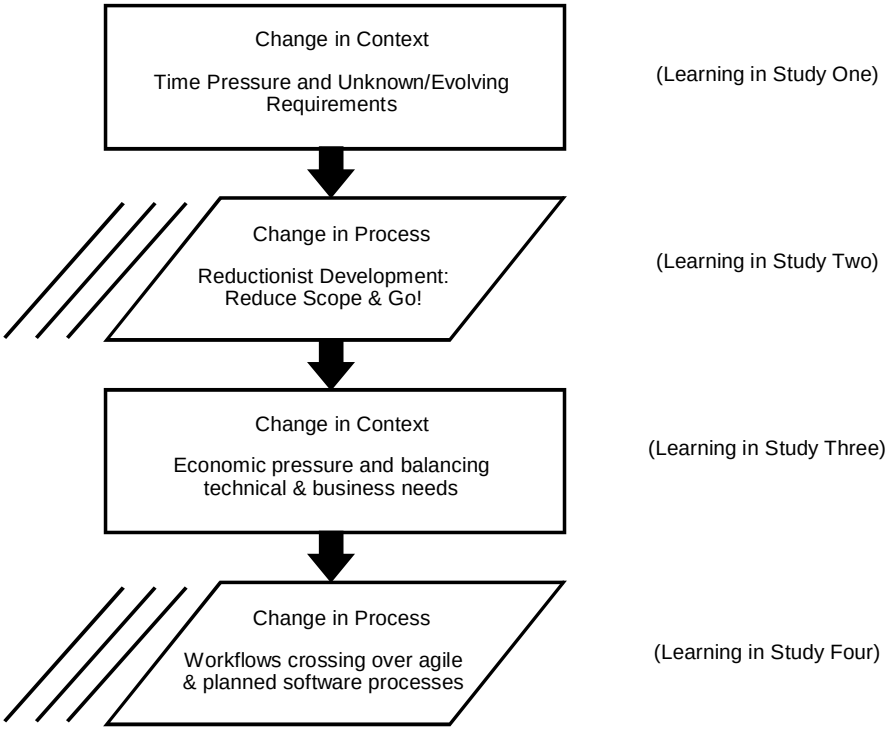

Fig. 5. Learning across the four studies.

From one perspective, this two-stage pattern is hardly surprising. A changing context undoubtedly drives changes in software processes. However, the solid evidence of an historical stage of fairly stable maturation of software process following a stage of more chaotic, context-driven process adaptation is surprising from a different perspective. It suggests that maturation in software processes may occur in historical cycles, rather than an endless progression of maturity-model driven advance. In other words, our evidence indicates that reoccurring periods of radically changing context will interrupt software process maturation.

Software process maturation does not necessarily restart from ground zero in each episode. Our evidence clearly indicates that our settings have learned from previous experience with new software processes. However, the evidence does suggest that overall global progress in software process maturity is episodic, with the possibility that each episode begins with a reversal, and then advances. It seems likely that each episodic advance brings the software discipline to an overall position of advancement. Thus software process progress is not steady, but characterized by episodes of decline and advance.

Our findings have a number of implications for theory and practice. First, the learning that we draw from across the four studies indicates that a broadening of the primarily local way we research and view software process maturity and ma-

turity models might be useful. Second, our findings show that practitioners have become so knowledgeable about agile development that they are able to use an agile approach beyond the initially recommended home ground and to successfully combine it with other development approaches and world views. Thus, it seems that the software development process has once again reached a state of some stability and maturity. Third, our findings show that a boundary spanning perspective is a useful theoretical lens for understanding the current success and stability of agile development.

References

Abrahamsson, P., Conboy, K., & Xiaofeng, W. (2009). 'Lots done, more to do': The current state of agile systems development research. *European Journal of Information Systems,* 18(4), 281-284.

Atlanta_Constitution. (2001, February 19). Internet growing by leaps and bytes: Study says americans trek to Cyberspace Is now a stampede. *The Atlanta Constitution,* pp. A1, A9.

Baskerville, R., Levine, L., Pries-Heje, J., Ramesh, B., & Slaughter, S. (2003). Is Internet-speed software development different? *IEEE Software,* 20(6), 70-77.

Baskerville, R., & Pries-Heje, J. (2001). Racing the e-bomb: How the Internet is redefining information systems development methodology. *Proceedings of the IFIP TC8/WG8.2 Working Conference on Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspective*, 49-68.

Baskerville, R., & Pries-Heje, J. (2002). Information systems development @ Internet speed: A new paradigm in the making. In S. Wrycza (Ed.), *Proceedings of the Tenth European Conference on Information Systems* (pp. 282-291). Gdansk: University of Gdansk.

Baskerville, R., & Pries-Heje, J. (2004). Short cycle time systems development. *Information Systems Journal,* 14(2), 237-264.

Beck, K. (2000). *Extreme programming explained: Embrace change.* Addison-Wesley.

Beck, K., & Fowler, M. (2001). *Planning extreme programming.* Boston: Addison-Wesley.

Boehm, B. (2002). Get ready for agile methods, with care. *IEEE Computer,* 35(1), 64-69.

Boehm, B., & Turner, R. (2004). *Balancing agility and discipline: A guide for the perplexed*. Boston: Addison-Wesley.

Cockburn, A. (2002). Learning from agile software development - Part one and two. *Crosstalk - The journal of Defence Software Engineering* (September and October).

Cox, A. (2005). What are communities of practice? A comparative review of four seminal works. *Journal of Information Science*, 31(6), 527-540.

Crosby, P. B. (1980). *Quality is free: The art of making quality certain.* New York: New American Library.

Cusumano, M., & Selby, R. (1995). *Microsoft secrets: How the world's most powerful company creates technology, shapes markets and manages people.* New York: Free Press.

Cusumano, M., & Yoffie, D. (2000). *Competing on Internet time: Lessons from Netscape and its battle with Microsoft*. New York: Touchstone.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information & Software Technology,* 50(9-10), 833-859.

Iansiti, M., & McCormack, A. (1997). Developing products on Internet time. *Harvard Business Review,* 75(5), 108-117.

Jeffries, R., Anderson, A., & Hendrickson, C. (2001). *Extreme programming installed*. Boston: Addison-Wesley.

Levina, N., & Vaast, E. (2005). The Emergence of boundary spanning competence in practice: Implications for implementation and use of information systems. *MIS Quarterly*, 29(2), 335-363.

Orlikowski, W. (1993). CASE tools as organizational change: Investigating incremental and radical changes in systems development. *MIS Quarterly*, 17(3), 309-340.

Pries-Heje, J., Baskerville, R., Levine, L., & Ramesh, B. (2005). The High Speed balancing game: How software companies cope with Internet speed. *Scandinavian Journal of Information Systems*, 16, 11-54.

Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. *IEEE Software*(July/Aug), 26-32.

Smith, P. G., & Reinertsen, D. G. (1995). *Developing products in half the time* (2nd Ed.). New York: Van Nostrand Reinhold.

Star, S. L., & Griesemer, J. R. (1989). Institutional ecology, `translations' and boundary objects: Amateurs and professionals in Berkeley's museum of Vertebrate Zoology, 1907-39. *Social Studies of Science, 19*, 387-420.

Strauss, A., & Corbin, J. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (2nd ed.). Thousand Oaks, Ca.: Sage.

Urquhart, C. (1997). Exploring analyst-client communication: Using grounded theory techniques to investigate interaction in informal requirements gathering. In A. S. Lee, J. Liebenau & J. I. DeGross (Eds.), *Information Systems and Qualitative Research* (pp. 149-181). London: Chapman and Hall.

Urquhart, C. (2000). Strategies for conversation and systems analysis in requirements gathering: A qualitative view of analyst-client communication. *The Qualitative Report (On-line serial)*, 4(1).

Author Biographies

Richard L. Baskerville is a Board of Advisors Professor of Information Systems and past chairman in the Department of Computer Information Systems, Robinson College of Business, Georgia State University. His research specializes in security of information systems, methods of information systems design and development, and the interaction of information systems and organizations. His interest in methods extends to qualitative research methods. Baskerville is the author of *Designing Information Systems Security* (J. Wiley) and more than 100 articles in scholarly journals, professional magazines, and edited books. He is Editor-in-Chief for *The European Journal of Information Systems* (EJIS) and serves on the editorial boards of *Business & Information Systems Engineering* (*Wirtschaftsinformatik*), and the *Information Systems Journal* (ISJ). A Chartered Engineer, Baskerville holds degrees from the University of Maryland (B.S. *summa cum laude*, Management), and the London School of Economics, University of London (M.Sc., Analysis, Design and Management of Information Systems, Ph.D., Systems Analysis).

Jan Pries Heje is Professor in Information Systems, Department of Communication, Business and IT, Roskilde University. Head of the User Driven IT-innovation Research Group. His research focuses on designing and building innovative solutions to managerial and organizational IT problems. Previous and current projects explore quality software development @ Internet speed, innovative capability in projects, the ability for an organization to improve, and how one can design a

process for making better sourcing decisions. Jan Pries-Heje is Past President of the Association of Information Systems in Scandinavia (IRIS). Jan Pries-Heje serves as the Danish National Representative to IFIP Technical Committee 8 on Information Systems where he is also Chair. Jan Pries-Heje has been Associate Editor for MIS Quarterly, ISJ, and EJIS. Jan Pries-Heje's research interests include project management, information systems development, and process improvement. He focuses on organisational and managerial issues.

Sabine Madsen (PhD) is Associate professor at the Department of Communication, Business, and IT at Roskilde University, Denmark. She has been employed as a project manager in the Danish IT-industry before pursuing an academic career. She completed her Ph.D. about process and method emergence in information system development practice in 2004. Her general research interests concern IS development processes, methods, and the relationship between research and practice and she has published on these topics in journals (such as EJIS, and ISJ), book chapters, and conference proceedings. She is a regular reviewer for IS journals and conferences. Recent research and teaching are in the areas of project management, outsourcing, and agile IS development. For these themes she is particularly interested in understanding and theorizing about the practices, challenges, and changes that IS developers and managers experience as a part of their day-to-day affairs.